الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التكوين والتعليم المهنيين
Ministère de la Formation et de l'Enseignement Professionnels

Centre National de l'Enseignement Professionnel à Distance (CNEPD)

المركز الوطني للتعليم المهني عن بعد (م.و.ت.م.ب)

COURS DE LANGUAGE PASCAL

SÉRIE 03

LES STRUCTURES DE CONTROLE

OBJECTIF PÉDAGOGIQUE : À la fin de ce cours, le stagiaire doit être capable d'utiliser les instructions important pour développer un programme.

PLAN DE LA LECON

I- LE CHOIX :

Le moment est venu de réaliser des programmes à l'aide d'instructions de contrôle qui vont autoriser, entre autre, des tests sur les opérateurs relationnels.

Nous allons étudier l'instruction de sélection IF qui active une alternative de traitement en fonction du résultat obtenu par l'évaluation d'une expression simple ou complexe.

1- Instruction IF... THEN ... ELSE :

Soit l'instruction conditionnelle suivante exprimée en algorithmique :

Si < Expression >

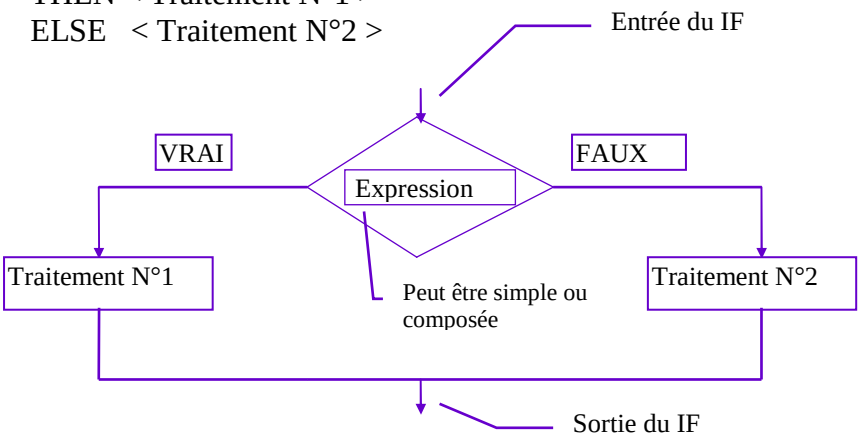
Alors < Traitement N°1 >

Sinon < Traitement N°2 >

Fsi ;

En langage PASCAL elle s'écrit sous la forme :

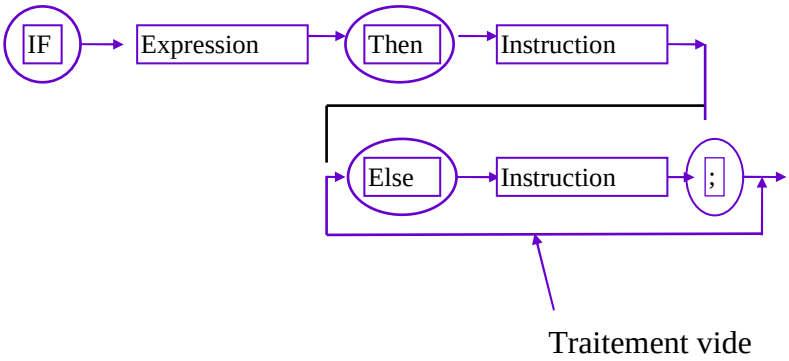
IF < Expression >
THEN < Traitement N°1 >
ELSE < Traitement N°2 >



On la représente par :

Il faut noter que < traitement N°2 > peut être vide.

Le diagramme de syntaxe de IF est :



Exemple :

Soit l'algorithme qui permet de déterminer la valeur supérieure entre deux valeurs entières A et B.

```
 ALGORITHME TEST ;

Début

A, B : Entier ;

Ecrire ('Entrer la valeur de la variable A') ;

Lire (A);

Ecrire ('Entrer la valeur de la variable B') ;

Lire (B) ;
Si A > B
 Alors       Ecrire (' A est supérieure a B') ;
Sinon       Si A < B
                Alors    Ecrire ('B est supérieure à A ')
                Sinon    Ecrire ('A est égale à B ') ;

                 Fsi ;
Fsi ;
```

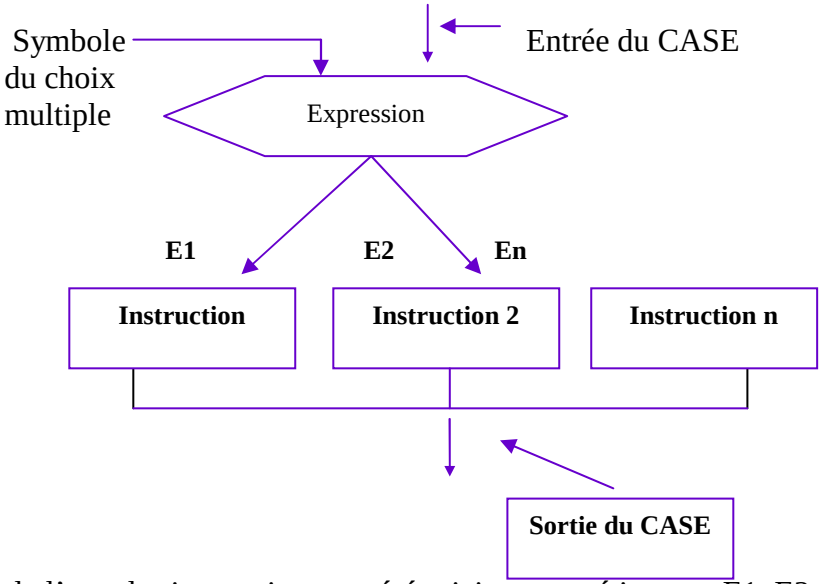
Le programme pascal correspondant est :

```
PROGRAM   TEST;
VAR       A, B: INTEGER;
BEGIN
  WRITELN ('Entrer la valeur de la  variable A') ;
  READLN (A) ;
  WRITELN ('Entrer la valeur de la  variable B') ;
   READLN (B);
   IF    A > B'
        THEN       WRITELN ('A  est supérieure à B')
         ELSE        IF    A < B
              THEN  WRITELN ('B est supérieure à A')
               ELSE      WRITELN ('A est égale  à B') ;
END.
```

2 - Instruction CASE ... OF...:

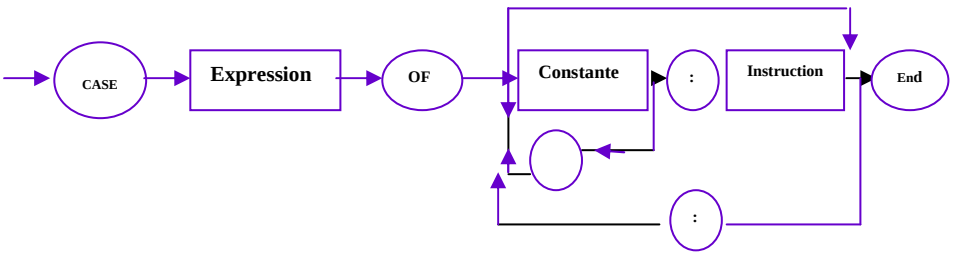
Pour une sélection simple, l'instruction IF permet d'opérer le choix entre deux alternatives mais, lors qu'il s'agit de choisir entre plusieurs alternatives, il faut utiliser l'instruction CASE ... OF..

Cette instruction a pour mot - clé CASE et elle est suivie d'une expression qui après évaluation, précise le traitement à effectuer. La liste des actions possibles est introduite par le mot -clé Of et se termine par END; Une des représentations symboliques peut être la suivante :



Seule l'une des instructions, repérées ici par une étiquette E1, E2, ... En est exécutée et dépend du résultat de l'évaluation de l'expression.

Le diagramme de syntaxe de l'instruction CASE est :



Exemple 1 :

```
PROGRAM  Qualité ;
VAR   i : INTEGER ;
BEGIN
WRITE ('ENTRER UNE VALEUR ENTRE 1 ET 4 :');
      READ (i)
      WRITELN;
      CASE i OF

                1: WRITELN ('Faible ');
                2 : WRITELN (' Moyen ') ;
                3 : WRITELN (' Bon ') ;
                4 : WRITELN (' Très bon ') ;
         END ;
      END.
```

Exécutions du programme 'Qualité':

1- Entrer une valeur entre 1 et 4 : 2
Moyen
2- Entrer une valeur entre 1 et 4 : 3
Bon

```
PROGRAM NMOIS ;
VAR    MOIS : INTEGER ;
BEGIN
  WRITE ('Entrer le numéro du mois ') ;
  READLN (mois);
  WRITELN;
  CASE   mois OF

        1 : WRITELN (' c'est Janvier ') ;
        2 : WRITELN (' c'est Février ') ;
        3 : WRITELN (' c'est Mars ') ;
        4 : WRITELN (' c'est Avril ') ;
        5 : WRITELN (' c'est Mai ') ;
        6 : WRITELN (' c'est Juin ') ;
```

```
            7 : WRITELN (‘ c’est Juillet ‘) ;
            8 : WRITELN (‘ c’est Août ‘) ;
            9 : WRITELN (‘ c’est Septembre’) ;
            10 : WRITELN (‘ c’est Octobre‘) ;
            11 : WRITELN (‘ c’est Novembre‘) ;
            12 : WRITELN (‘ c’est Décembre‘) ;

        END;
    END.
```

Remarque :

Dans le cas où on ne peut pas énumérer toutes les valeurs de <expression>, il existe une autre variante de l’instruction CASE ... OF ... qui est :

```
CASE <Expression> OF
        Constante1 : Instruction1 ;
        Constante2 : Instruction2 ;


        Constante n : Instruction n ;
        OTHERWISE : Instruction j ;
END ;
```

II- LA RÉPÉTITION OU L'ITÉRATION :

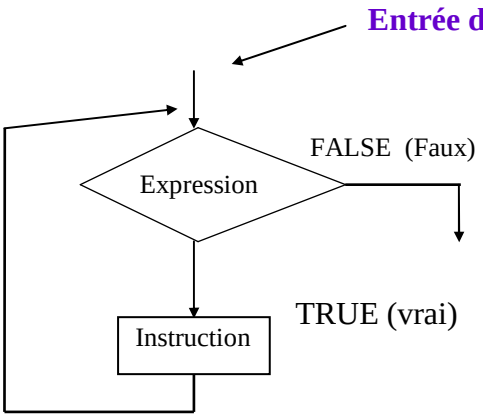
D'une manière générale, il existe deux façons de répéter des instructions :

- D'une manière inconditionnelle (on dit aussi définie) les instructions sont répétées un nombre de fois bien déterminé lorsque l'on aborde la répétition. Ce type de répétition est réalisé en pascal par l'instruction FOR ... TO ... DO ...
- D'une manière conditionnelle (on dit aussi indéfinie) la poursuite ou l'abondons de la répétition sont conditionnés par ce qui se passe au sein de la boucle. Nous verrons plus précisément que cela s'exprime à travers une expression booléenne (conditions) .En PASCAL, deux instructions (assez semblables) permettent de réaliser ce type de répétition : REPEAT ... UNTIL qui répète des instructions jusqu'à ce qu'une condition soit réalisée, WHILE ... DO ... qui répète des instructions tant qu'une certaine condition soit vraie.

WHILE (tant que)	Expression (expression booléenne)	**DO** (Faire)	Instruction (suite d'instructions)		
REPEAT (répéter)	instruction (suite d'instructions)	**UNTIL** (jusqu'à)	Expression (expression booléenne)		
FOR (pour)	expression 1 (variable := valeur ou paramètre)	**TO** (à) (ordre croissant)	Expression 2 (expression booléenne)	**DO** Faire	Instruction (suite d'instructions)
FOR (pour)	Expression (Variable := valeur ou paramètre)	**DOWNTO** (à) (ordre décroissant)	Expression 2 (expression booléenne)	**DO** Faire	Instruction (suite d'instructions)

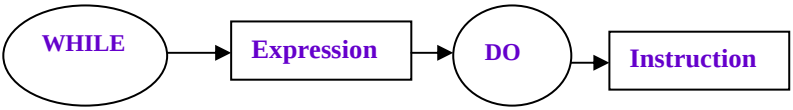
1- Instruction while ... DO ... :

La première de ces instructions réalise un traitement tant qu'une condition est satisfaite ; elle est détachée par les mots - clés : WHILE et DO .Voici la représentation symbolique de WHILE ... DO :



Dans ce schéma nous constatons que le test de <Expression>s'effectue avant le traitement : Tant que l'évaluation de l'expression est vérifiée on poursuit l'exécution de <Instruction>, sinon il y a rupture de l'exécution et sortie du WHILE.

Diagramme de syntaxe :



Si l'expression n'est pas satisfaite la première fois le traitement ne sera jamais exécutée.

Remarque :

Si <Instruction> est un ensemble d'instructions alors cet ensemble est limité par les mots réservés BEGIN et END.

Exemple 1 :

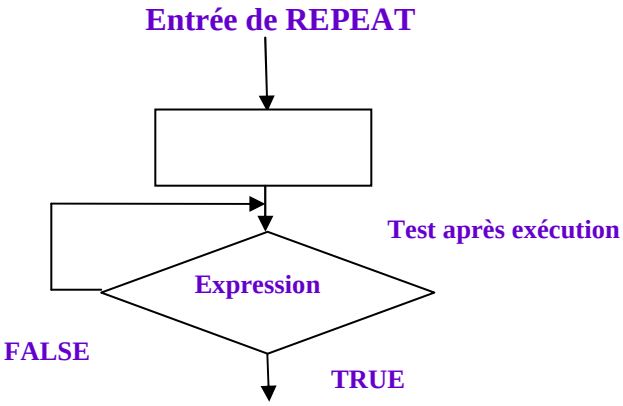
PROGRAM EXEMPL1 ;
VAR I,S : INTEGER ;
BEGIN
I : = 0 ; S : = 0 ;
WHILE I <= 100 DO
BEGIN
S = S + I ;
I = I + 1 ;
END ;
WRITELN ('SOMME=' ; S) ;
	Résultat de l'exécution
	
	SOMME = 5050

Exemple 2 :

PROGRAM EXEMPL2 ;
VAR SOMME, NOMBRE : INTEGER ;
BEGIN
SOMME : = 0 ;
WHILE SOMME <= 100 DO
BEGIN
WRITE ('Donner un nombre')
READLN (NOMBRE) ;
SOMME :=SOMME+NOMBRE ;
END ;
WRITELN ('SOMME OBTENUE = ', SOMME) ;
END.
	exécution
	
	Donner un nombre : 15
	Donner un nombre : 50
	Donner un nombre : 43
	
	SOMME OBTENUE= 108

2- Instruction repeat .... UNTIL :

Alors que l'instruction précédente donne le test d'itération en début d'opération, l'instruction REPEAT.... donne le test en fin de traitement tel que le montre la représentation symbolique ci-après :



Il y a au moins une fois exécution de l'instruction puisque le test est affecté en fin d'action.

Aussi, l'instruction REPEAT...UNTIL constitue l'une des deux manières de réaliser une répétition conditionnelle

Exemple :

PROGRAM EXEMPL3 ;
VAR N : INTEGER;
BEGIN
REPEAT
WRITE (Donner un entier positif ;');
READLN (N) ;
UNTIL N > 0 ;
WRITELN ('MERCI pour ',N) ;
END.
	exécution
	
	Donner un entier positif ;-1
	Donner un entier positif : 0
	Donner un entier positif : 5
	Merci pour 5

On ne sait pas à priori combien de fois la « boucle » sera ainsi répétée.

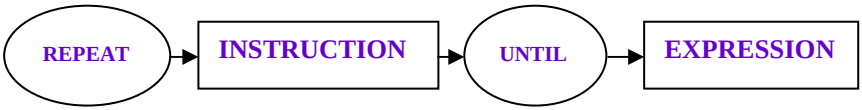
Néanmoins de par sa nature un fois, l'expression booléenne (ici N > 0) est alors évaluée. Si elle est vraie, on passe à l'instruction suivant la boucle (on dit alors qu'on sort la boucle). Si la condition est fausse, on parcoure à nouveau toutes les instructions de la boucle avant d'évaluer une nouvelle fois la condition et ainsi se suite.

Ne confondez pas WHILE ... DO et REPEAT... UNTIL.

N'oubliez pas :

- **WHILE .... DO provoque un test avant l'action**
- **REPEAT... UNTIL provoque un test après l'action**

Diagramme de syntaxe REPEAT... UNTIL:



Exemple :

PROGRAM EXEMP4 ;
VAR N : INTEGER;
BEGIN
REPEAT
WRITE (Donner un entier positif ');
READLN (N) ;
WRITELN ('MERCI ') ;
UNTIL N >0
WRITELN ('MERCI pour ',N) ;
END.
	exécution
	
	Donner un entier positif ;-1
	merci
	Donner un entier positif : 0
	merci
	Donner un entier positif : 5
	Merci
	Merci pour 5

Exemple :

PROGRAM EXEMPL5 ;
VAR S, I : INTEGER ;
BEGIN
S : = 0 ; I : = 1 ;
REPEAT
S : = S + I ;
I : = I + 1 ;
UNTIL I<=200
WRITE (‘ le résultat = ’, S) ;
END.
	exécution
	
	Le résultat = 20100

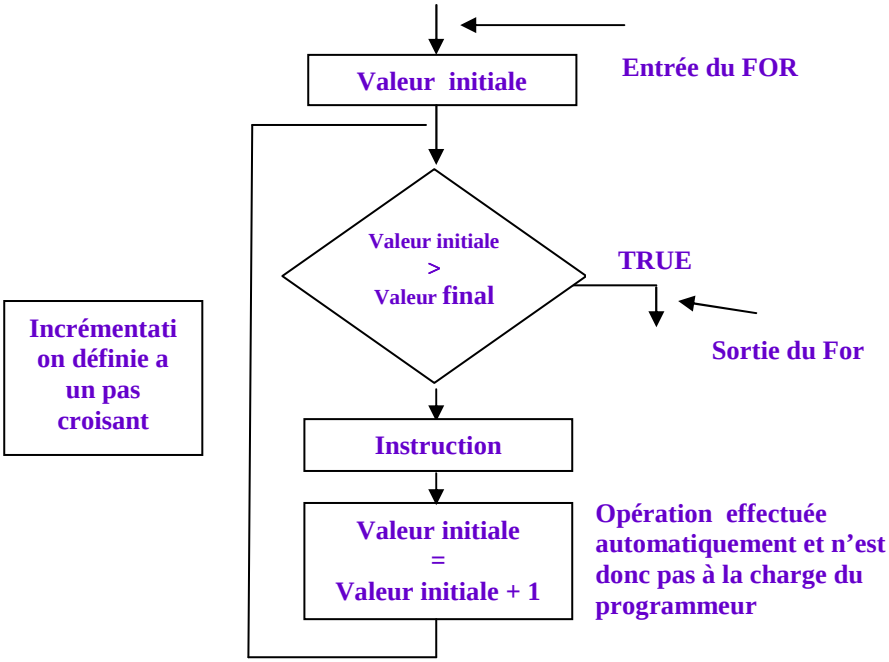
L’itération cesse lorsque la condition est satisfaite, les instructions en REPEAT et UNTIL sont exécutées au moins une fois. REPEAT et UNTIL servent de parenthèses, BEGIN et END ne sont pas nécessaires.

3- Instruction FOR...TO (DOWNTO)... DO :

Il existe deux autres façons pour résoudre des traitements répétitifs consistant à donner la limite inférieure ou supérieure du nombre d’itérations à effectuer. Il s’agit des instructions FOR ... TO ... DO qui précise la limite inférieure.

Dans le premier cas l’incrémentation est définie à un pas croissant, dans le deuxième cas à un pas décroissant.

Schémas symboliques de la boucle FOR :



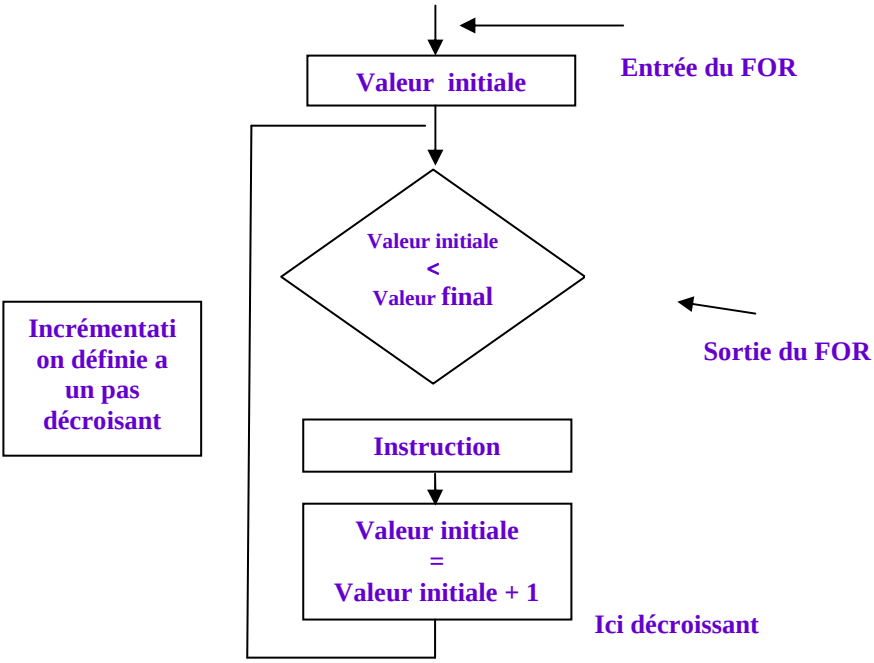
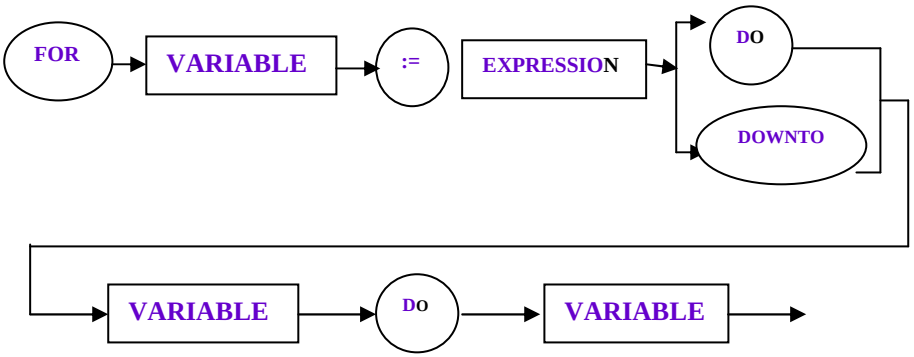


Diagramme de syntaxe de l'instruction FOR :



Remarque :

1- La valeur initiale d'expression1 indique la valeur à partir de laquelle le traitement répétitif doit commencer et la valeur finale d'expression2 donne la limite à ne pas dépasser.

2- Dans le cas ou la valeur initiale est égale à la valeur finale, aucun traitement n'est effectué.

3- FOR ... TO ... DO précise qu'il s'agit d'un parcours en ordre croissant.
FOR ... DOWNTO ... DO ... précise qu'il s'agit d'un parcours en ordre décroissant.

Exemples :

```
PROGRAM  COMPT0 ;
  VAR  N : INTEGER ;
  BEGIN
    FOR N :=1TO8
     DO
 WRITE (N,'a pour double',2* N);
END.
```

```
1 a pour double 2
2 a pour double 4
3 a pour double 6
4 a pour double 8
5 a pour double 10
6 a pour double 12
7 a pour double 14
8 a pour double 16
```

```
PROGRAM  COMPTP ;
  VAR   N : INTEGER ;
 BEGIN
   FOR  N := 8  DOWNTO 1 DO
   WRITELN  (  N,'  a  pour  double'
,2* N) ;
  END.
```

```
8 a pour double 16
7 a pour double 14
6 a pour double 12
5 a pour double 10
4 a pour double 8
3 a pour double 6
2 a pour double 4
1 a pour double 2
```

PROGRAM ALPHABET ; VAR C : CHAR ; BEGIN FOR C : = ‘a’ TO ‘Z’ DO WRITE (C) ; END.	abcdefghijklmnopqrstuvwxyz

4- Instruction GOTO ... :

En théorie, les structures de contrôle proposées par PASCAL sont plus que suffisantes pour traduire toutes les situations possibles.

Cependant PASCAL dispose également d’une instruction de branchement inconditionnel GOTO. En voici un exemple.

_____
_____

IF (erreur) THEN GOTO CAS-ERREUR

_____
_____
_____

CAS-ERREUR : _____
_____

ERREUR est supposée être une variables booléenne, l’exécution de l’instruction IF provoque un «branchement» à l’instruction précédée par «l’étiquette» CAS ERREUR, lorsque la variable erreur possède la valeur vrai,

Les étiquettes ainsi utilisées doivent être obligatoirement déclarées dans une instruction LABEL figurant dans la partie déclaration dans notre cas, ce serait

LABEL CAS-ERREUR. Ces étiquettes sont formées soit d'un identificateur soit d'un nombre de 1 à 4 chiffres.

Une étiquette permet d'identifier toute instruction du programme et de pouvoir y accéder par sa désignation à l'aide de l'instruction GOTO

Diagramme de syntaxe de LABEL :

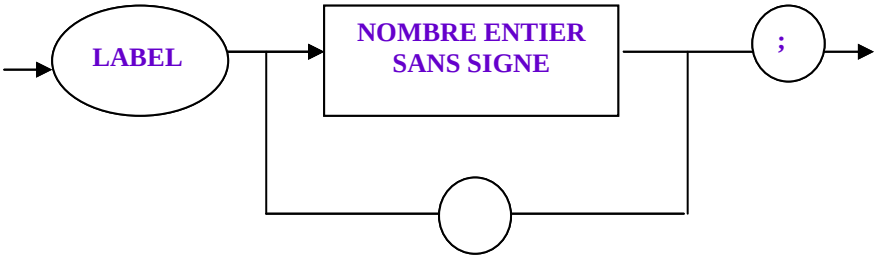
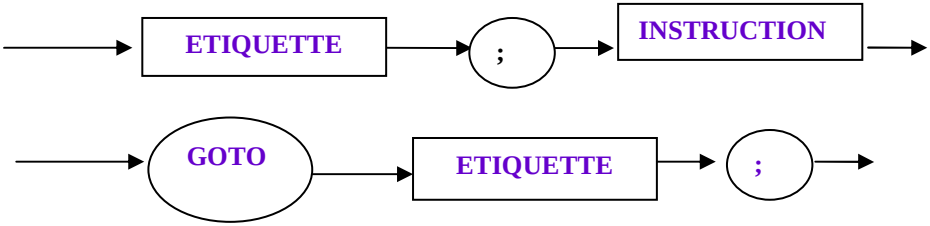


Diagramme de syntaxe GOTO :



Exemple :

```
PROGRAM  GT ;
LABEL   1, 2, FIN ;
VAR  SUITE: CHAR ;
BEGIN
        1 : WRITELN (‘ 1 ère étiquette) ;
        2 : WRITELN (‘2 ème étiquette) ;
            GOTO  FIN ;
 FIN : WRITELN (‘FIN de programme ‘) ;
        WRITELN (‘voulez-vous continuer (0/n)’) ;
         READ ( SUITE ) ;
         IF SUITE = ‘0’ THEN  GOTO  1
 END.
```

RÉSUMÉ :

Nous avons vu que la représentation d’un programme consiste à utiliser une suite d’actions structurées, car l’ordre dans lequel les actions doivent être exécutées est fondamental. Certaines actions sont élémentaires, d’autres sont plus complexes et permettent de représenter les divers enchaînements.

Cette leçon a été consacrée pour les structures de contrôle qui sont nécessaires dans un programme à l’évaluation des expressions logiques pour l’enchaînement à telle ou telle directive selon la valeur trouvée.

En Pascal, il existe des structures conditionnelles de choix telles que: IF ... THEN ... ELSE, CASE ... OF ... et des structures repetitives telles que: WHILE ... DO, REPEAT ... UNTIL ..., FOR ... TO (DOWNTO) ... DO et GOTO ... .

Ce sont donc, un moyen de construire des actions plus ou moins complexes.

EXERCICES D'APPLICATION :

EXERCICE N°01 :

Ecrire un programme qui permet le calcul des carrés des N premiers nombres entiers. Le programme doit permettre l'affichage des carrés de tous les nombres entiers dont la limite est saisie au clavier.

EXERCICE N°02 :

Ecrire un programme qui donne la somme des N premiers nombres entiers avec N valeur entière positive entrée au clavier.

EXERCICE N° 03 :

Réaliser un programme qui permet de résoudre l'équation du premier degré ax+b=0 ou a et b sont des réels.

EXERCICE N° 04 :

Ecrire un programme qui recherche le plus grand de deux nombre entiers.

EXERCICE N° 05 :

Elaborer un programme qui affiche la forme d'un nombre binaire : l'équivalent du chiffre décimal entré au clavier par l'utilisateur.

EXERCICE N°06 :

Soit deux nombres entiers x et n ($n \geq 0$) calculer X^n par multiplications successives sans utiliser l'opérateur (élever à la puissance) .

EXERCICE N°07 :

Si on s'intéresse à la somme des inverses des nombres entiers :

$$1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+...+\frac{1}{p}+\frac{1}{p+1}+...$$

On sait qu'elle « diverge » vers l'infini. Autrement dit, quel que soit le nombre réel positif A, il existe toujours un entier n tel que :

$$1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{n}>A \qquad (1)$$

Ecrire un programme auquel on fournit en donnée, une valeur de A (borne inférieure) et qui déterminera le plus grand entier n satisfaisant à la condition (1)

EXERCICE N° 08 :

Ecrire un programme qui Calcule la moyenne des notes fournies au clavier avec un «dialogue » se présentant ainsi :

Combien de notes : 4

NOTE 1 : 12
NOTE 2 : 15.25
NOTE 3 : 13.50
NOTE 4 : 8.75

Moyenne des 4 notes : 12.37

CORRIGÉS DES EXERCICES :

EXERCICE N°01 :

```
PROGRAM  CARRES ;
 VAR     I, N, CARRE: INTEGER;
  BEGIN
     WRITE (‘ Donner un nombre :’) ;
     READLN (N)
     WRITELN (‘Voilà les carres: ‘);
      FOR   I: = 1 TO N
            DO   BEGIN   CARRE: = SQR (I);
                                 WRITE (CARRE);
                 END;
   END.
```

EXERCICE N° 02:

```
PROGRAM SOMME;
  VAR     I, N, T: INTEGER;
  BEGIN
     T : =0 ;
     WRITE (‘ Donner un nombre N’,) ;
      READLN (N);
      FOR    I: = A   DOWNTO 1
               DO    T : =  T +  I ;
      WRITELN (‘La Somme des ‘, N,’ premiers nombres
entiers = :‘, t) ;
END.
```

EXERCICE N°03:

```
PROGRAM EQUATION;
   VAR A, B, X:   REAL;

 BEGIN
WRITELN (‘ Résolution de l’équation AX+B = 0 ‘) ;
```

```
WRITELN (‘ Donner les valeurs de A et B ‘) ;
 READLN (A, B);
 IF A = 0
     THEN     IF  B = 0
        THEN   WRITE (‘ Tout nombre réel est solution ‘)
        ELSE     WRITE (‘ Pas de racine ‘)
     ELSE      BEGIN
                 X : = - B/A ;
                 WRITE (‘ Il existe une racine =’, X) ;
               END;
 END.
```

EXERCICE N° 04:

```
PROGRAM GRAND2V;
VAR      A, B: INTEGER;
BEGIN
  WRITELN (‘Détermination du plus grand de deux nombres’) ;
  WRITE (‘Donner deux nombre entiers quelconques ‘ ) ;
   READLN (A, B);
   IF A > B
      THEN     WRITELN (‘A est plus grand que B ‘)
       ELSE      IF B > A
           THEN    WRITELN (‘B est plus grand que A’)
                   ELSE    WRITELN (‘A est EGAL a B’);
    END.
```

EXERCICE N° 05:

```
        PROGRAM EQUIV;
          X: INTEGER;
          BEGIN
            WRITELN (‘ Conversion Décimale Binaire’) ;
             WRITE (‘Donner un chiffre décimal’) ;
              CASE X     OF
```

```
            0   :  WRITELN ('0000');
            1   :  WRITELN ('0001');
            2   :  WRITELN ('0010');
            3   :  WRITELN ('0011');
            4   :  WRITELN ('0100');
            5   :  WRITELN ('0101');
            6   :  WRITELN ('0110');
            7   :  WRITELN ('0111');
            8   :  WRITELN ('1000');
            9   :  WRITELN ('1001');
                   OTHERWISE

    WRITELN ('ERREUR ce n'est pas un chiffre décimal ') ;
                END;
              END.
```

EXERCICE N° 06 :

```
  PROGRAM  PUISSANCE ;
   VAR:  X, N, P: INTEGER;
   BEGIN
     WRITELN (' Taper un nombre = ') ;
      READLN (X) ;
      WRITE (' Taper la puissance supérieure à ZERO ') ;
      READLN (N);
      WHILE N < 0
                DO    BEGIN
        WRITELN (' Vous faite erreur ') ;
                WRITE ('Taper la puissance supérieure à ZERO ') ;
                        READLN (N);

              END;
      P: = 1;
      WHILE N > 0
                DO    BEGIN
```

```
                P : = P * X ;
                N : = N - 1;
              END ;
    WRITELN (‘ La puissance de ce nombre = ‘, P’) ;
  END.
```

EXERCICE N° 07:

```
PROGRAM SERIE;
 VAR I: INTEGER;
 LIMITE, SOMME : REAL ;
 BEGIN
   WRITE (‘Donner la valeur de la borne inférieure de la somme’) ;
   WRITE (‘ 1 + 1/2 + 1/3 +...+ 1/n comprise entre 0 et 1’) ;
   READLN (LIMITE);
   SOMME: = 0;
   I: = 0;
   REPEAT
   I: = I + 1;
   SOMME: = SOMME + 1/I;
   UNTIL  SOMME > LIMITE ;
   WRITELN (‘ La valeur de n vérifiant la condition est :’, I) ;
  END.
```

EXERCICE N° 08 :

```
PROGRAM  MOYENNE ;
 VAR   NOTE, SOMME, MOYENNE : REAL ;
         I, NB-NOTES:  INTEGER;
 BEGIN
   REPEAT
     WRITE (‘Combien de notes : ‘) ;
     READLN (NB-NOTES) ;
```

```
      UNTIL NB-NOTES > 1;
       SOMME: = 0;
       FOR   i: = 1   TO NB-NOTES
             DO   BEGIN
                    WRITE ('NOTES', I);
                     READLN (NOTE);
                      SOMME := SOMME + NOTE ;
                   END;
       MOYENNE := SOMME /NB-NOTES ;
       WRITELN ;
       WRITELN ('Moyenne des ', NB-NOTES, 'NOTES :
', MOYENNE )
  END.
```